

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L8	35	"virtual machine manager" and "common language runtime" and "java virtual machine"	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 13:56
L9	33851	"707/".cccls.	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 13:56
L10	26	"virtual machine manager" and "common language runtime" and "java virtual machine" and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:00
L11	871	("virtual machine manager" or "common language runtime" or "java virtual machine") and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:07
L12	18	((("virtual machine manager" or "common language runtime" or "java virtual machine") same (manipulated or controlled)) and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:19
L13	4	((("virtual machine manager" or "common language runtime") same (manipulated or controlled)) and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:20
L14	109	("virtual machine manager" or "common language runtime") and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:20
L15	26	("virtual machine manager" and "common language runtime") and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:22
L16	35	("virtual machine manager") and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:21
L17	100	("common language runtime") and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:22
L18	67	("common language runtime") and manip\$ and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:23
L19	2	("common language runtime") same manip\$ and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 14:23
L20	67	("common language runtime") and manip\$ and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 16:15
L21	16	("java virtual machine" or JVM) same manipulation and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 15:23

EAST Search History

L22	29	("java virtual machine" or JVM) same manipulats and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 15:20
L23	0	("virtual machine manager" or vmm) same manipulation and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 15:23
L24	1	("virtual machine manager" or vmm) same manipulats and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 15:32
L25	45	("virtual machine manager" or vmm) and manipulats and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 15:33
L26	100	("common language runtime") and L9	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 16:15
L27	10	("common language runtime") and L9 not microsoft	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 16:22
L28	61	("common language runtime" or CLR) and L9 not microsoft	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 16:20
L29	10	("common language runtime") and L9 not microsoft	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/21 16:23
L30	10	("common language runtime") and L9 not microsoft	US-PGPUB; USPAT; USOCR; EPO; DERWENT	OR	ON	2006/04/21 16:23
S1	12	((("20020198891") or ("20020091702") or ("5900870") or ("6047291") or ("6108004") or ("6112024") or ("6199195") or ("6338056") or ("6370541") or ("6519597") or ("6556983") or ("6578046"))).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/18 11:22
S2	1	("20050055354").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/19 15:46
S3	4156	"707/100".cccls.	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 15:48
S4	33725	"707/".cccls.	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 15:48
S5	15565	S4 and item	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 15:49

EAST Search History

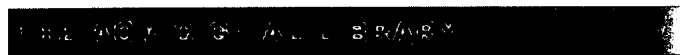
S6	0	S4 and "plurality of items"	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 15:50
S7	1078	S4 and (plurality adj2 item)	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 15:50
S8	550	S4 and (plurality adj item)	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 15:50
S9	27	S4 and (plurality adj item) and (item adj folder)	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 16:57
S10	1	("6343287").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/18 17:02
S11	0	("objectwithpropertywithapplication withprogram").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/18 17:05
S12	346	object with property with application with program	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 17:06
S13	91	(object with property with application with program)and S4	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 17:35
S14	2300	((object adj2 oriented) with database)and S4	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 17:39
S15	47	((object adj2 oriented) with database with group)and S4	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/18 17:39
S16	1	("20030196052").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/19 10:27
S17	2633	(default or "trash can") with (folder or directory)	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:47
S18	33725	"707/".ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:47
S19	627	(default or "trash can") with (folder or directory) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:49
S20	8	(default or "trash can") with (folder or directory)with delete and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:53

EAST Search History

S21	1	(move adj3 (default or "trash can")) with (folder or directory) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:54
S22	1	(move adj4 (default or "trash can")) with (folder or directory) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:55
S23	6	(move with (default or "trash can")) with (folder or directory) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:55
S24	38	((move or store) with (default or "trash can")) with (folder or directory) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:55
S25	13	((move or store) adj5 (default or "trash can")) with (folder or directory) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 15:57
S26	3	recycle adj3 (delete or deletion) and S18	US-PGPUB; USPAT; USOCR	OR	ON	2006/04/19 16:08
S27	3	recycle adj3 (delete or deletion) and S18	US-PGPUB; USPAT; USOCR; EPO; DERWENT	OR	ON	2006/04/19 16:09
S28	0	default adj3 ((delete or deletion) with move) and S18	US-PGPUB; USPAT; USOCR; EPO; DERWENT	OR	ON	2006/04/19 16:09
S29	1	((("2005004994") or ("20050050054"))).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/19 17:33
S30	0	((("2005004994") or ("20050050054"))).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/19 17:28
S31	2	((("20050049994") or ("20050050054"))).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/04/19 17:33


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

 Terms used **common language runtime**

Found 111 of 175,083

Sort results by

☒ [Save results to a Binder](#)
[Try an Advanced Search](#)

Display results

☒ [Search Tips](#)
[Try this search in The ACM Guide](#)
☐ Open results in a new window

Results 1 - 20 of 111

 Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [next](#)

 Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Formalization of generics for the .NET common language runtime](#)



Dachuan Yu, Andrew Kennedy, Don Syme

 January 2004 **ACM SIGPLAN Notices , Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '04**, Volume 39 Issue 1

Publisher: ACM Press

Full text available: pdf(171.28 KB)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

We present a formalization of the implementation of generics in the .NET Common Language Runtime (CLR), focusing on two novel aspects of the implementation: mixed specialization and sharing, and efficient support for run-time types. Some crucial constructs used in the implementation are dictionaries and run-time type representations. We formalize these aspects type-theoretically in a way that corresponds in spirit to the implementation techniques used in practice. Both the techniques and the form ...

Keywords: .NET, CLR, generics, polymorphism, run-time types

2 [Technical correspondence: Language integration in the common language runtime](#)



Jennifer Hamilton

 February 2003 **ACM SIGPLAN Notices**, Volume 38 Issue 2

Publisher: ACM Press

Full text available: pdf(974.52 KB)

 Additional Information: [full citation](#), [abstract](#), [references](#)

The Common Language Runtime (CLR) is language and platform-neutral, and provides the underlying infrastructure for the Microsoft .NET Framework. A key innovation in the CLR is its support for multiple programming languages, enabling programming language integration at the runtime level to a much greater degree than is currently possible.


Keywords: common type system, exception handling, intermediate language, language interoperability, metadata, virtual machine

3 [Design and implementation of generics for the .NET Common language runtime](#)



Andrew Kennedy, Don Syme

 May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5

Publisher: ACM PressFull text available:  [pdf\(1.25 MB\)](#)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

The Microsoft.NET Common Language Runtime provides a shared type system, intermediate language and dynamic execution environment for the implementation and inter-operation of multiple source languages. In this paper we extend it with direct support for parametric polymorphism (also known as generics), describing the design through examples written in an extended version of the C# programming language, and explaining aspects of implementation by reference to a prototype extension to the runtime ...

4 [Inter-language object sharing with the common language runtime: infrastructure for MS.NET](#)

Jennifer Hamilton

July 2001 **Proceedings of the 23rd International Conference on Software Engineering****Publisher:** IEEE Computer SocietyFull text available:  [pdf\(93.97 KB\)](#)Additional Information: [full citation](#), [index terms](#) [Publisher Site](#)

5 [Manipulating managed execution runtimes to support self-healing systems](#)



Rean Griffith, Gail Kaiser

May 2005 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2005 workshop on Design and evolution of autonomic application software DEAS '05**, Volume 30 Issue 4**Publisher:** ACM PressFull text available:  [pdf\(346.02 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Self-healing systems require that repair mechanisms are available to resolve problems that arise while the system executes. Managed execution environments such as the Common Language Runtime (CLR) and Java Virtual Machine (JVM) provide a number of application services (application isolation, security sandboxing, garbage collection and structured exception handling) which are geared primarily at making managed applications more robust. However, none of these services directly enables applications ...

Keywords: CLR, JIT compiler, autonomic computing, common language runtime, online reconfiguration, self-healing systems

6 [Java Virtual Machine: JVM versus CLR: a comparative study](#)

Jeremy Singer

June 2003 **Proceedings of the 2nd international conference on Principles and practice of programming in Java PPPJ '03****Publisher:** Computer Science Press, Inc.Full text available:  [pdf\(84.98 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#)

We present empirical evidence to demonstrate that there is little or no difference between the Java Virtual Machine and the .NET Common Language Runtime, as regards the compilation and execution of object-oriented programs. Then we give details of a case study that proves the superiority of the Common Language Runtime as a target for imperative programming language compilers (in particular GCC).

7 [Adventures in interoperability: the SML.NET experience](#)



Nick Benton, Andrew Kennedy, Claudio V. Russo

August 2004 **Proceedings of the 6th ACM SIGPLAN international conference on Principles and practice of declarative programming****Publisher:** ACM PressFull text available: pdf(434.04 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

SML.NET is a compiler for Standard ML that targets the Common Language Runtime and is integrated into the Visual Studio development environment. It supports easy interoperability with other .NET languages via a number of language extensions, which go considerably beyond those of our earlier compiler, MLj. This paper describes the new language extensions and the features of the Visual Studio plugin, including syntax highlighting, Intellisense, continuous type inference and debugger support. We dis ...

Keywords: applications of declarative programming, functional programming, integration of paradigms, programming environments

8 RaceTrack: efficient detection of data race conditions via adaptive tracking



Yuan Yu, Tom Rodeheffer, Wei Chen

October 2005 **ACM SIGOPS Operating Systems Review , Proceedings of the twentieth ACM symposium on Operating systems principles SOSP '05**, Volume 39 Issue 5**Publisher:** ACM PressFull text available: pdf(321.34 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bugs due to data races in multithreaded programs often exhibit non-deterministic symptoms and are notoriously difficult to find. This paper describes RaceTrack, a dynamic race detection tool that tracks the actions of a program and reports a warning whenever a suspicious pattern of activity has been observed. RaceTrack uses a novel hybrid detection algorithm and employs an adaptive approach that automatically directs more effort to areas that are more suspicious, thus providing more accurate war ...

Keywords: race detection, virtual machine instrumentation

9 Typing a multi-language intermediate code



Andrew D. Gordon, Don Syme

January 2001 **ACM SIGPLAN Notices , Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '01**, Volume 36 Issue 3**Publisher:** ACM PressFull text available: pdf(257.05 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

The Microsoft .NET Framework is a new computing architecture designed to support a variety of distributed applications and web-based services. .NET software components are typically distributed in an object-oriented intermediate language, Microsoft IL, executed by the Microsoft Common Language Runtime. To allow convenient multi-language working, IL supports a wide variety of high-level language constructs, including class-based objects, inheritance, garbage collection, and a security mechanism b ...


10 Modeling methodology b: Open source initiatives for simulation software: multi-language, open-source modeling using the microsoft .NET architecture



Richard A. Kilgore

December 2002 **Proceedings of the 34th conference on Winter simulation: exploring new frontiers****Publisher:** Winter Simulation Conference

Full text available: Additional Information:

 [pdf\(311.36 KB\)](#)
[full citation](#), [abstract](#), [references](#), [citations](#)

This presentation reports on the opportunities and limitations Microsoft .Net architecture for supporting the development of a common, open-source, multi-language platform for simulation software support. While the paper supporting the presentation focuses on the underlying foundation within the .Net architecture, the conference presentation represents an important milestone in the OpenSML project corresponding to the first release of a common library supporting the C#, VB.Net and Java/J# lan ...

11 [Parametric polymorphism for software component architectures](#)



Cosmin E. Oancea, Stephen M. Watt

October 2005 **ACM SIGPLAN Notices , Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '05**, Volume 40 Issue 10

Publisher: ACM Press

Full text available:  [pdf\(375.78 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Parametric polymorphism has become a common feature of mainstream programming languages, but software component architectures have lagged behind and do not support it. We examine the problem of providing parametric polymorphism with components combined from different programming languages. We have investigated how to resolve different binding times and parametrization semantics in a range of representative languages and have identified a common ground that can be suitably mapped to different lan ...

Keywords: antiunification, generics, parametric polymorphism, software component architecture, templates

12 [MyCoG.NET: towards a multi-language CoG toolkit](#)



A. Paventhan, Kenji Takeda

November 2005 **Proceedings of the 3rd international workshop on Middleware for grid computing MGC '05**

Publisher: ACM Press

Full text available:  [pdf\(4.12 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Grid application developers may exploit Commodity Grid Toolkits (CoG) to readily consume Globus Grid services. Existing CoG Toolkits are language-specific and are available for Java, Python, and the Matlab scripting environment. In this paper we describe the first multi-language CoG toolkit, MyCoG.NET, based around the Microsoft NET Framework. MyCoG provides set of classes and APIs for .NET. We demonstrate its programmability using FORTRAN, C++, C# and Java, and discuss its performanc ...

Keywords: .NET, CLR, CoG, GRAM, GSSAPI, GridFTP


13 [CodeBricks: code fragments as building blocks](#)



Giuseppe Attardi, Antonio Cisternino, Andrew Kennedy

June 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation PEPM '03**, Volume 38 Issue 10

Publisher: ACM Press

Full text available:  [pdf\(294.34 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We present a framework for code generation that allows programs to manipulate and generate code at the source level while the joining and splicing of executable code is carried out automatically at the intermediate code/VM level. The framework introduces a

data type Code to represent code fragments: methods/operators from this class are used to reify a method from a class, producing its representation as an object of type Code. Code objects can be combined by partial application to other Code ob ...

Keywords: domain specific language, generative programming, metaprogramming, multistage programming, program generation, program transformation, reflection

14 A framework for obfuscated interpretation

Akito Monden, Antoine Monsifrot, Clark Thomborson

January 2004 **Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32 CRPIT '04**

Publisher: Australian Computer Society, Inc.

Full text available:  pdf(357.56 KB) Additional Information: [full citation](#), [abstract](#), [references](#)

Software protection via obscurity is now considered fundamental for securing software systems. This paper proposes a framework for obfuscating the program interpretation instead of obfuscating the program itself. The obfuscated interpretation enables us to hide functionality of a given program P unless the interpretation being taken is revealed. The proposed framework employs a finite state machine (FSM) based interpreter to give the context-dependent semantics to each instruction in P ...

Keywords: encryption, obfuscation, software protection



15 Industrial sessions: database internals - II: Hosting the .NET Runtime in Microsoft

SQL server

Alazel Acheson, Mason Bendixen, José A. Blakeley, Peter Carlin, Ebru Ersan, Jun Fang, Xiaowei Jiang, Christian Kleinerman, Balaji Rathakrishnan, Gideon Schaller, Beysim Sezgin, Ramachandran Venkatesh, Honggang Zhang

June 2004 **Proceedings of the 2004 ACM SIGMOD international conference on Management of data**

Publisher: ACM Press

Full text available:  pdf(249.27 KB) Additional Information: [full citation](#), [abstract](#), [references](#)

The integration of the .NET Common Language Runtime (CLR) inside the SQL Server DBMS enables database programmers to write business logic in the form of functions, stored procedures, triggers, data types, and aggregates using modern programming languages such as C#, Visual Basic, C++, COBOL, and J++. This paper presents three main aspects of this work. First, it describes the architecture of the integration of the CLR inside the SQL Server database process to provide a safe, scalable, secure, an ...

16 Continuations from generalized stack inspection



Greg Pettyjohn, John Clements, Joe Marshall, Shriram Krishnamurthi, Matthias Felleisen

September 2005 **ACM SIGPLAN Notices , Proceedings of the tenth ACM SIGPLAN international conference on Functional programming ICFP '05**, Volume 40 Issue 9

Publisher: ACM Press

Full text available:  pdf(213.07 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Implementing first-class continuations can pose a challenge if the target machine makes no provisions for accessing and re-installing the run-time stack. In this paper, we present a novel translation that overcomes this problem. In the first half of the paper, we introduce a theoretical model that shows how to eliminate the capture and the use of first-class continuations in the presence of a generalized stack inspection mechanism. The second half of the paper explains how to translate this mode ...

Keywords: A-normal form, continuation-passing style, continuations, defunctionalization, scheme, stack inspection, web programming

17 Move to component based architectures: introducing Microsoft's .NET platform into the college classroom



Meg Murray

January 2004 **Journal of Computing Sciences in Colleges**, Volume 19 Issue 3

Publisher: Consortium for Computing Sciences in Colleges

Full text available:  pdf(45.40 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

A transformation has been occurring in the architectural model for computer-based application intense software systems. This new model, software-as-a-service, will have a profound impact on the design and development of software for many years to come and as such college level computing curriculums will need to incorporate the concepts and methodologies associated with this new architecture. The platform is built upon a view of interrelated, distributed peer-level software modules and components ...

18 Intermediate representation engineering: Porting legacy interpretive bytecode to the CLR



Jeremy Singer

June 2002 **Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002 PPPJ '02/IRE '02**

Publisher: National University of Ireland

Full text available:  pdf(427.88 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

We describe and evaluate five different approaches to executing legacy BCPL programs on the Common Language Runtime. The approaches cover the entire spectrum of machine-independent compilation, from high-level source code to low-level bytecode. We discuss the trade-offs involved with each approach.


19 Weaving Ada 95 into the .net environment



 Martin C. Carlisle, Ricky E. Sward, Jeffrey W. Humphries

December 2002 **Proceedings of the 2002 annual ACM SIGAda international conference on Ada: The engineering of correct and reliable software for real-time & distributed systems using Ada and related technologies**

Publisher: ACM Press

Full text available:  pdf(254.58 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper explains our efforts to add Ada to Microsoft's family of .NET languages. There are several advantages to weaving Ada into the Common Language Environment provided by the .NET environment. This paper explains our approach and current progress on the research. We provide the means to extract Ada specification files from Microsoft Intermediate Language (MSIL) code and compile Ada programs into MSIL.

Keywords: Ada 95, Microsoft .Net environment, common language runtime, just-in-time compiling

20 HPC.NET - are CLI-based Virtual Machines Suitable for High Performance Computing?



Werner Vogels

November 2003 **Proceedings of the 2003 ACM/IEEE conference on Supercomputing**

Publisher: IEEE Computer Society

Full text available:  [pdf\(163.35 KB\)](#) Additional Information: [full citation](#), [abstract](#)

The Common Language Infrastructure is a new, standardized virtual machine that is likely to become popular on several platforms. In this paper we review whether this technology has any future in the high-performance computing community, for example by targeting the same application space as the Java-Grande Forum. We review the technology by benchmarking three implementations of the CLI and compare those with the results on Java virtual machines.

Results 1 - 20 of 111

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)